



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

| APPLICATION NO.          | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|--------------------------|-------------|----------------------|---------------------|------------------|
| 09/892,989               | 06/27/2001  | Shashidhar P. Jade   | 206871              | 3769             |
| 23626                    | 7590        | 11/19/2003           | EXAMINER            |                  |
| LEYDIG VOIT & MAYER, LTD |             |                      | VO, TED T           |                  |
| 6815 WEAVER ROAD         |             |                      | ART UNIT            |                  |
| ROCKFORD, IL 61114-8018  |             |                      | PAPER NUMBER        |                  |

2122

DATE MAILED: 11/19/2003

5

Please find below and/or attached an Office communication concerning this application or proceeding.

**Office Action Summary**

Application No.

09/892,989

Applicant(s)

JADE ET AL.

Examiner

Ted T. Vo

Art Unit

2122

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 27 June 2001.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-31 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-31 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on \_\_\_\_\_ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
- 11) ☐ The proposed drawing correction filed on \_\_\_\_\_ is: a) ☐ approved b) ☐ disapproved by the Examiner.  
If approved, corrected drawings are required in reply to this Office action.
- 12) ☐ The oath or declaration is objected to by the Examiner.

**Priority under 35 U.S.C. §§ 119 and 120**

- 13) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).  
a) ☐ All b) ☐ Some \* c) ☐ None of:  
1. ☐ Certified copies of the priority documents have been received.  
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.  
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).  
\* See the attached detailed Office action for a list of the certified copies not received.
- 14) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).  
a) ☐ The translation of the foreign language provisional application has been received.
- 15) ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892) 4) ☐ Interview Summary (PTO-413) Paper No(s). \_\_\_\_\_
- 2) ☒ Notice of Draftperson's Patent Drawing Review (PTO-948) 5) ☐ Notice of Informal Patent Application (PTO-152)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449) Paper No(s) 4. 6) ☐ Other:

**DETAILED ACTION**

1. This action is in response to the application filed on 06/27/2001.

Claims 1-31 are pending in the application.

***Claim Rejections - 35 USC § 103***

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

A person shall be entitled to a patent unless –

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

A. Claims 1-12, 15-31 is rejected under 35 U.S.C. 103(a) as being unpatentable over Dmitri, "Spying on COM Objects", Windows Developer's Journal 7-1999.

Given the broadest reasonable interpretation of followed claims in light of the specification:

As per claim 1:

Dmitri discloses intercepting and tracing messages by spying on COM objects. Dmitri discloses a tracer that injects a piece of code ('*patch*') into an address space of a target application. When this code is executed, it changes the pointer to load a tracing DLL. Debugger code would be used to inject breakpoints so that, when a breakpoint is triggered, Execution transfers control to a function that performs interception process (See page 3, lines 15-27, referring line 24). A Spy DLL is also injected so that the debugger could change control to execute it (see page 3, lines 44-50). The tracer has ability to intercept

values ('*graphic primitives and attributes*') and information that are returned from COM interface (see page 3, lines 7-13, 'The tracer should be prepared...').

The teaching covers the claim limitation: "***A method for capturing one or more graphic primitives and attributes associated with such graphic primitives of a display object, the method comprising the steps of:***

***injecting a spy component*** (page 3, line 46, 'inject the spy DLL') ***into a target process*** (page 3, line 46, 'searches the target process') ***by a calling process*** (Tracer), ***wherein the spy component is an executable program module*** (spy DLL);

***capturing*** (see page 1, section, Full Text, first paragraph, 'Intercepting and tracing message and function calls', and see page 3, lines 53-54, 'to intercept functions in newly loaded DLLs') ***the one or more graphic primitives and attributes associated with such graphic primitives*** (interception process) ***during the execution of the target process; and***

***returning the graphic primitives and attributes associated with such graphic primitives to the calling process***" (page 3, lines 3-13, referring "return a COM interface", "values returned through the function's parameters", "returning interfaces in the configuration files").

Dmitri does not address "***graphic primitives and attributes***" as recited in "***capturing the one or more graphic primitives and attributes***"; and "***returning the graphic primitives and attributes associated with such graphic primitives***". Dimitri discloses capturing the COM interface functions in debugging Win32 application using API, where COM objects are parts of graphic applications which are parts of Win32 application. COM objects are the components, used for building the windows including graphics (Page 1, section Full Text, first, second and third paragraphs, particularly see first paragraph).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include, "***graphic primitives and attributes***", as particularly functional elements of COM interface for graphics debugging, where graphics is also part of windows.

The modification would be obvious because one of ordinary skill in the art would be motivated for taking advantage of spying DLLs which is provided for capturing all COM interface functions of COM objects in

API Win32 debugging, for covering particularly functions directly to graphics matters of window applications.

As per claim 2: Dmitri discloses, "***The method of claim 1, wherein the step of injecting the spy component is performed by an injection component*** (page 3, line 46, 'blnjectAgent()') ***that is invoked by the calling process*** (Tracer).

As per claim 3: Dmitri discloses, "***The method of claim 2, wherein the injection component is an executable program module comprising executable routines for injecting source code into a target process***" (page 3, lines 46-47, 'writes an agent to it').

As per claim 4: Dmitri discloses, "***The method of claim 1, wherein the step of injecting includes inserting one or more function patches*** (page 3, line 18, 'injects a tiny piece of code', or line 24, 'inject a breakpoint') ***into one or more executable program modules that correspond to an operating system*** (page 3, line 20, 'API functions') ***upon which the target process is being executed, the executable program modules having instructions for rendering graphics primitives to a graphical user interface***" (Page 3, lines 3-13, "return a COM interface", "values returned through the function's parameters", "returning interfaces in the configuration files").

As per claim 5: Dmitri discloses, "***The method of claim 4, wherein the step of inserting the function patches is performed by the spy component***" (see page 3, line 46, 'inject the spy DLL using blnjectAgent()', and see line 58, 'vHookModule() inside the spy DLL').

As per claim 6: Dmitri discloses, "***The method of claim 1, wherein the step of injecting includes installing one or more hook functions*** (see page 3, see line 58, 'vHookModule() inside the spy DLL') ***into the operating system APIs*** (page 3, line 20, 'API functions') ***that generate system messages in***

***the event of a display object being output to a user interface screen*** (see page 1, section Full Text, first paragraph, 'tracing messages') ***during the execution of the target process***".

As per claim 7: Dmitri discloses, "***The method of claim 6, wherein the step of installing is performed by the spy component***" (see page 3, see line 58, 'vHookModule()' inside the spy DLL').

As per claim 8: Dmitri discloses, "***The method of claim 6, wherein the system messages provide context information that is descriptive of an invoked action within the target process*** (see page 1, section Full Text, first paragraph, 'tracing messages and function calls').

As per claim 9: Dmitri discloses a debugger that does hooking management. The debugger injects breakpoints for transferring control to tracing DLLs (see page 3, lines, 24-27, 'inject a breakpoint). The debugger manages hooks inside an injected DLL (see page 3, line, 58, 'vHookModule()'). The debugger associates hooking with a spy DLL (see page 3, line 58, 'vHookModule()' inside the spy DLL'). This teaching covers the limitation: "***The method of claim 6, wherein the one or more hook functions (vHookModule()) are installed by a hook management component ('debugger/tracer') that is called upon by the spy component after injection into the target process***" (see page 3, see line 58, 'vHookModule()' inside the spy DLL', and see page 4, lines 1-2, 'the DLL\_PROCESS\_ATTACH event', and 'bHookAllModules()').

As per claim 10: Dmitri discloses the debugger that removes breakpoints (page 3, line 56-57). Spy DLLs and Hooking are incorporated with debugging so that after completing an interception, the debugger returns control to execute the application normally (uninstall) (see page 3, lines 58-60). This teaching covers limitation: "***The method of claim 9, wherein the hook management component is responsible for uninstalling the one or more hook functions upon termination of the target process***" (see page 3, lines 58-60, 'control returns to the DLL's original entry point').

As per claim 11: Dmitri discloses, "***The method of claim 1, wherein the step of capturing includes calling the one or more hook functions to intercept*** (see page 4, lines 4-5, 'vHookModule() is used by debugger code to intercept new DLLs') ***any system messages generated as a result of an invoked action within the target process***" (see page 1, section Full Text, first paragraph, 'tracing messages and function calls').

As per claim 12: Within one address space, Dmitri discloses a tracer that injects a piece of code ('function patches') into the address space to perform tracing (page 3, line 18, 'inject a tiny piece of code'). Within the application, Dmitri provides a set of trace groups; each trace group is set by a bit flag to enable or disable tracing (see page 10, line 7, 'The bit flag is different for different group of trace operator'). This discloses, "***The method of claim 11, wherein the one or more hook functions set a flag to activate the one of the function patches installed by the spy component***"

As per claim 15: Dmitri discloses, "***The method of claim 1, wherein the step of returning includes sending the context information captured by the one or more hook functions to the calling process as a system message***" (see page 4, lines 4-5).

As per claim 16: Dmitri addresses the interception process in tracing (page 3, lines 26-27). The returns from the interception are the returned values, interfaces, functions, COM interface, etc (see page 3, lines 3-13). The trace results a performance of interception including the execution of spy DLLs. Injecting a small piece of code is also associated with injecting spy DLLs ('function patches') (see page 3, lines 44-50). Since Spy DLLs are injected via breakpoints, each spy DLL acts like a patch function to capture COM interfaces (page 4, lines 4-5). The teaching covers the claim limitation: "***wherein the step of returning further includes sending the graphics primitives and attributes associated with such graphics primitives*** ('values, interfaces, functions, COM interface') ***that are captured by the one or more function patches*** (page 4, lines 4-5, vHookModule()) ***to the calling process as a system message***" (see page 1, section Full Text, first paragraph, 'Intercepting and tracing messages').

As per claim 17: Independent claim 17 recites a computer-readable medium having computer-executable instructions for capturing one or more graphic primitives and attributes associated with such graphic primitives of a display object. The computer-executable instructions perform the steps having claimed functionality corresponding to the claim 1. Claim 17 is rejected in the same reason as set forth in connection to the rejection of claim 1.

As per claim 18:

Dmitri discloses, "***an injection component for injecting a spy component into a target process residing on a computer*** (page 3, line 46, 'inject the spy DLL').

Dmitri discloses, "***a spy component*** (page 3, line 46, 'the spy DLL') ***for capturing*** (see page 1, section, Full Text, first paragraph, 'Intercepting and tracing message and function calls', and see page 3, lines 53-54, 'to intercept functions in newly loaded DLLs') ***graphics primitives and attributes associated with such graphics primitives*** (interception process) ***in connection with system messages that are generated by the target process as a result of an invoked action within the target process*** (page 1, section Full Text, first paragraph); and

Dmitri discloses a debugger that is able to implement breakpoints for injecting spy DLLs (see page 3, lines 44-47, 'target process hits a breakpoint', and 'the best time to inject the spy DLL'). The debugger is able to remove breakpoints after intercepting the interface information (see page 3, 55-60, 'The debugger removes the breakpoint'). The hook functions are implemented with the spy DLLs (see page 3, line 58, 'vhookModule() inside the spy DLL', see page 4, 'vHookModule() is used by the debugger code'). The functionality of injecting/removing breakpoints used by the debugger covers the limitation: "***a hook management component*** ('Tracers'; 'Debugger') ***for installing and uninstalling*** (injecting/removing breakpoints) ***one or more hook functions*** (vHookModule() incorporated inside Spy DLL)(page 3, line 58) ***into one or more program modules that are executed by an operating system residing on the computer*** (page 3, line 20, 'API functions'), ***the program modules having instructions for generating system messages during the execution of the target process.***"

Dmitri does not address ***“graphics primitives and attributes associated with such graphics primitives”*** as recited in ***“a spy component for capturing graphics primitives and attributes associated with such graphics primitives in connection with system messages that are generated by the target process as a result of an invoked action within the target process”***.

However COM interface functions in API Win32 are parts of Graphics applications, and ***graphic primitives and attributes*** are elements of graphics interface.

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include, ***“graphics primitives and attributes associated with such graphics primitives”***, particularly functional elements of COM interface for graphics debugging, where graphics is also part of windows.

The modification would be obvious because one of ordinary skill in the art would be motivated for taking advantage of spying on COM objects, provided for capturing all COM interface functions of COM objects in API Win32 debugging, for covering particularly functions directly to graphics matters of window applications.

As per claim 19: Dmitri discloses, ***“The system of claim 18, wherein the injection component is an executable program module consisting of executable instructions for injecting the spy component into the executable code of the target process”*** (page 3, line 48, “LoadLibriray() to load the spy DLL”).

As per claim 20: Dmitri discloses, ***“The system of claim 18, wherein the injection component*** (page 3, line 46, ‘using blnjectAgent()’) ***injects the spy component into the target process on behalf of a calling process”*** (See page 4, line 2, ‘bHookAllModules’, (or Debugger)).

As per claim 21: Dmitri discloses, ***“The system of claim 20, wherein the calling process is a computer executable application”*** (bHookAllModule() or ‘Debugger’ appears to be a computer executable application).

As per claim 22: Dmitri discloses, "***Th system of claim 18, wherein the spy component inserts on or more function patches*** (page 3, lines 46-47, 'bInjectAgent() searches the target process address space for a writable page and writes an agent to it') ***into one or more executable program modules that correspond to the operating system upon which the target process is being executed, the executable program modules having instructions for rendering graphics primitives to a graphical user interface*** (page 3, lines 3-13, referring "return a COM interface", "values returned through the function's parameters", "returning interfaces in the configuration files").

As per claim 23: Claim 23 is further limitation of claim 22, where Dmitri teaches claim 22 limitation as addressed.

Dmitri discloses, "***capture*** (see page 3, lines 3-13) ***graphics primitives and associated attributes of such graphics primitives that are rendered to a user interface*** ('values, interfaces, functions, COM interface') ***by a display object as a result of an action invoked within the target process***". Dmitri discloses capturing values, interfaces, functions (page 3, lines 3-13) that are intercepted from the interception process. The interception process is also included in execution of spying agents (see page 3, lines 44-54).

As per claim 24: Claim 24 is further limitation of claim 23, where Dmitri teaches claim 23 limitation as addressed.

Dimitri discloses, "***where spy component packages the graphics primitives and attributes associated with such graphics primitives*** ('values, interfaces, functions, COM interface') ***and sends it to the calling process as a system message***" (see page 1, section Full Text, first paragraph, 'Intercepting and tracing messages and function calls is very important for debugging application under windows').

As per claim 25: Dmitri discloses, "***The system of claim 18, wherein the spy component calls the hook management component*** (page 4, lines 1-5, 'bHookAllModules() is called') ***to insert one or more hook functions into one or more executable program modules that correspond to the operating***

***system upon which the target process is being executed*** (page 4, lines 1-5, 'vHookModules() is used by the debugger code').

As per claim 26: Within one address space, Dmitri discloses a tracer that injects a piece of code ('function patches') into the address space to perform tracing (page 3, line 18, 'inject a tiny piece of code'). Within the application, Dmitri provides a set of trace groups; each trace group is set by a bit flag to enable or disable tracing (see page 10, line 7, 'The bit flag is different for different group of trace operator').

This teaching covers limitation. ***"The system of claim 25, wherein the one or more hook functions set a flag to activate the one or more function patches installed by the spy component"***

As per claim 27: Dmitri discloses, ***"The system of claim 25, wherein the one or more hook functions intercept any system messages*** (see page 4, lines 4-5, and see page 1, section Full Text, first paragraph, 'Intercepting and tracing messages and function calls') ***generated as a result of an invoked action within the target process"***.

As per claim 28: Dmitri discloses, ***"The method of claim 27, wherein the system messages*** (see page 1, section Full Text, first paragraph, 'Intercepting and tracing messages') ***contain context information that is descriptive of an invoked action within the target process*** (page 3, lines 3-13, referring to the returns of 'values, interfaces, information about functions, COM interface function').

As per claim 29: Dmitri discloses, ***"The system of claim 18, wherein the spy component packages the context information*** ('values, interfaces, information about functions, COM interface function') ***and sends it to the calling process as a system message"*** (see page 1, section Full Text, first paragraph, 'Intercepting and tracing messages and function calls') ***and sends it to the calling process as an OS message*** (see page 1, section Full Text, first paragraph, referring to 'trace messages and API function calls').

As per claim 30: "Dmitri discloses, ***"The system of claim 18, wherein the hook management component is responsible for installing*** (page 3, line 24, 'debugger code will inject a breakpoint') ***and uninstalling*** (page 3, lines 55-57, 'The debugger removes the breakpoint') ***hook functions on behalf of the spy component"***.

(Examiner interprets that the debugger has the means of hook management because Dmitri teaching of injection or removal of a breakpoint is associated with an injected spy DLL. When the breakpoint is injected, a spy DLL is also injected (page 3, lines 44-45). The spy DLL also includes hook functions for the interception process (page 3, lines 457-58)).

As per claim 31: Dmitri discloses, ***"The system of claim 30, wherein the hook functions are uninstalled by the hook management component upon termination of the target process*** (see page 3, line 59-60, 'control returns to DLL's original entry point').

B. Claims 13-14 are rejected under 35 U.S.C. 103(a) as being unpatentable over Dmitri, "Spying on COM Objects", Windows Developer's Journal 7-1999, and further in view of Ouaknine et al. (US 6,091,422).

As per claim 13: Dmitri discloses capturing messages and function calls from intercepting and tracing COM objects (page 1, section Full Text, first paragraph), where messages include the returned values or interfaces of COM interfaces (page 3, lines 3-13). This is associated with invoking an action in the target process (page 3, lines 55-56, 'target application hits this new breakpoint'). This teaching covers: ***"capturing includes invalidating a display object that is output to a user interface as a result of the invoked action within the target process"***.

Dmitri does not explicitly disclose, ***"invalidating a display object"*** recited in ***"capturing includes invalidating a display object that is output, to a user interface as a result of the invoked action within the target process"***.

Ouaknine discloses, "**invalidating a display object**" (Ouaknine: see column 8, lines 56-64, 'Invalidated portions may be blanked in a display'). Ouaknine discloses a rendering process that allows update data of a rendering portion to invalidate another portion. The invalidation allows only the update portion to be displayed in the rendering process (see column 8, lines 59-63).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include the teaching of Ouaknine in rendering an update portion by invalidating another portion in graphics display into the teaching returning interceptions using spy DLLs of Dmitri.

The modification would be obvious because one of ordinary skill in the art would be motivated to utilize a common use of using portion invalidation in graphics rendering. Doing so would confirm to a requirement of graphics rendering if allowing only necessary data to be displayed.

As per claim 14: Claim 14 is further limitation of claim 13, where Dmitri and Ouaknine in combination teach claim 13 limitation as addressed.

With further limitation of claim 14, Dmitri discloses calling hooks, injecting breakpoints in the target application for interception process (Dmitri: page 3, lines 26-27) The interception captures the returned values, interfaces, functions, COM interfaces, etc (Dmitri: page 3, lines 3-13). The interception is also performed by a patching spy DLL ('more function patches') from injecting code (Dmitri: page 3, line 47).

Dmitri does not explicitly address "**invalidating include calling the function patches**" as recited in: "*The method of claim 13, wherein the step of invalidating includes calling the function patches to capture the graphics primitives and attributes associated with such graphics primitives as they are drawn to the user interface to render the display object*".

Ouaknine discloses, "**invalidating**" (Ouaknine: see column 8, lines 56-64, 'invalidated by update data') for allowing updated data of a rendering portion to invalidate another portion. The invalidation allows only the update portion to display in the rendering process (see column 8, lines 59-63).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to include the teaching of invalidating in Ouaknine and Dmitri in combination with further teaching of Ouaknine in invalidating graphics rendering.

Art Unit: 2122

The modification would be obvious because one of ordinary skill in the art would be motivated to take advantage invalidation as a common use of a rendering process. Doing so would confirm to a requirement of graphics rendering if allowing only wanted data returned from interception.

***Conclusion***

3. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

**Harel**, US No. 6,064,381, discloses a method for identifying human difficulties in operating a computerized system

**Ward et al.**, US No. 6,314,470, discloses a system included with a hook management for attaching diagnostic tool in a graphic execution.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (703) 308-9049. The examiner can normally be reached on Monday-Friday from 8:00 AM to 5:30 PM ET. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Dam, can be reached on (703) 305-4552.

The fax phone numbers:

(703) 872-9306 (for formal communication intended for entry);

(703) 746-5429 (for informal or draft communication, please label "PROPOSED" or "DRAFT").

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-3900.

TED T. VO

Patent Examiner

Art Unit: 2122

November 10, 2003